

# Existential Graphs

Computability and Logic

# Existential Graphs

- A graphical logic system developed by C.S. Peirce almost 100 years ago.
- Peirce studied semiotics: the relationship between symbols, meanings, and users.
  - Peirce stressed the power of iconic representations
  - Existential Graphs allow the user to express logical statements in a completely graphical way.
    - Alpha (Propositional Logic)
    - Beta (Predicate Logic)
    - Gamma (Modal Logic)

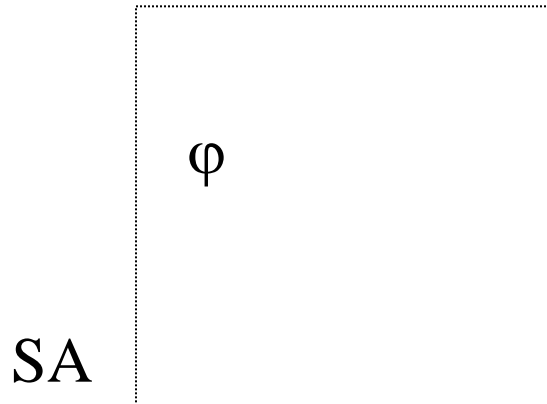
# Alpha

- Alpha is the part of Existential Graphs (EG) corresponding to propositional (or truth-functional) logic (PL).
- This presentation covers:
  - symbolization
    - from PL to EG
    - from EG to PL
  - inference
    - rules
    - Strategies

## *Symbolization*

# Sheet of Assertion

- To assert some statement in EG, you put the symbolization  $\varphi$  of that statement on a sheet of paper, called the ‘Sheet of Assertion’ (SA). Thus, to assert the truth of some statement  $p$ , draw:

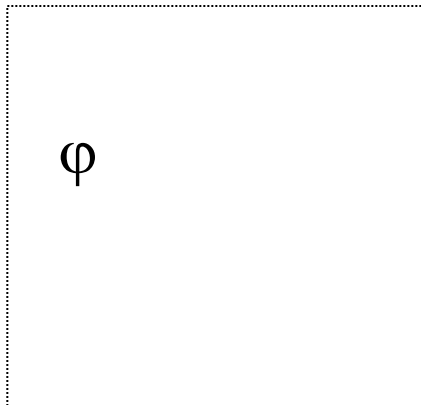


where  $\varphi$  is the  
symbolization  
of  $p$

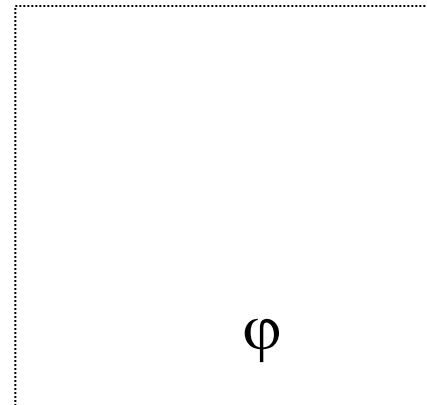
## *Symbolization*

# Location is irrelevant

- The location of the symbolization on the SA does not matter: as long as it is *somewhere* on the SA, it is being asserted. Thus:



states the  
same as:



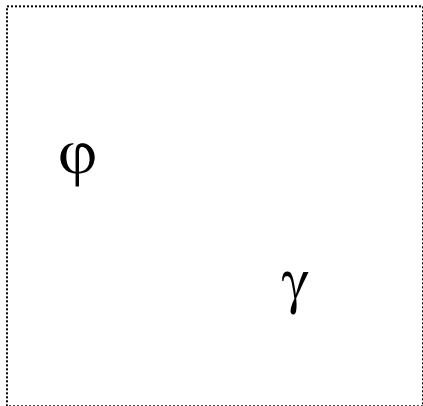
- In fact, the above two graphs are regarded as completely identical.

## *Symbolization*

# Juxtaposition and Conjunction

- By drawing the symbolization of two statements on the SA, you are asserting the truth of both statements at once. Hence, the mere *juxtaposition* of two symbolizations on the SA can be interpreted as the assertion of a single *conjunction*.

Thus:



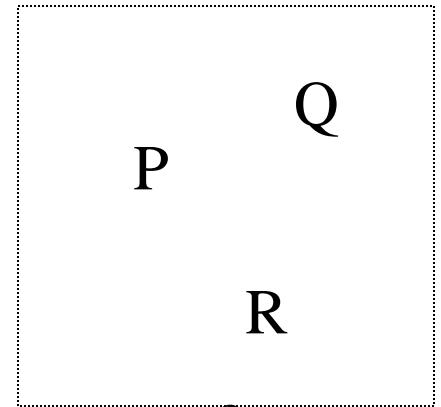
can be seen as the assertion of both  $\varphi$  and  $\gamma$ , but also as the assertion of  $\varphi \wedge \gamma$ .

## *Symbolization*

# **Generalized Conjunction**

- Since any number of symbolizations can be juxtaposed on the SA, juxtaposition becomes a kind of generalized conjunction that can have any number of conjuncts. Moreover, since the location of each of the symbolizations on the SA does not matter, no particular order on these conjuncts is imposed. This coincides with our abstract understanding of conjunction, and it is here that EG has an important advantage over the linear notation of traditional PL. An example will help:

## *Symbolization*



## **Generalized Conjunction: Example**

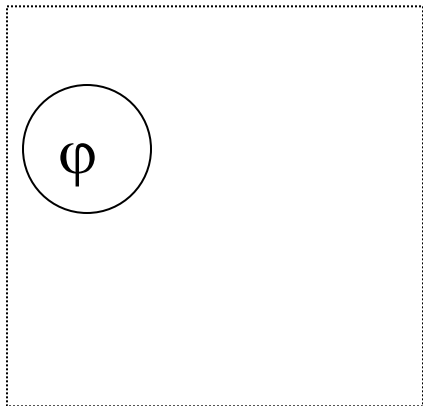
- The top-right graph can be interpreted in any of the following ways in PL:
  - the assertion of 3 statements: P, Q, and R
  - the assertion of 2 statements: P and  $Q \wedge R$ 
    - (or of R and  $P \wedge Q$ , or of P and  $R \wedge Q$ , etc.)
  - the assertion of a single statement:  $P \wedge (Q \wedge R)$ 
    - (or of  $(P \wedge Q) \wedge R$ , or of  $(Q \wedge R) \wedge P$ , or of  $P \wedge (R \wedge Q)$ , etc.!)
- However, our abstract understanding is in each case the same: P, Q, and R are all, and at the same time, true. Hence, a single symbolization should suffice, and this is exactly what EG can offer us.



## *Symbolization*

# Cut and Negation

- You assert the negation of some statement by drawing a cut (circle, oval, rectangle, or any other enclosing figure) around the symbolization of that statement. Thus:



asserts that  $\varphi$  is false.

(from now on, the SA will no longer be drawn)

## *Symbolization*

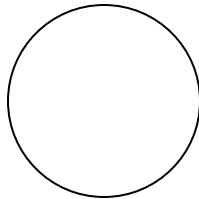
# **Empty Graph and Tautology**

- Any blank piece of paper can be seen as an ‘empty graph’. Thus, any graph can be seen as the juxtaposition of that graph with an empty graph. However, since this juxtaposition should express the same as the original graph, any empty graph expresses a tautology. Another way of looking at this is to view any tautology as an ‘empty claim’ since, being a tautology, it effectively doesn’t claim anything at all.

*Symbolization*

## **Empty Cut and Contradiction**

- A cut without any contents is called an ‘empty cut’. Since an empty cut negates an empty graph, any empty cut expresses a contradiction ( $\perp$ ).



## *Symbolization*

# **Expressive Completeness**

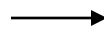
- Using juxtaposition for conjunction, and cuts for negation (and letters for simple, atomic statements), any compound, truth-functional statement can be symbolized in EG. That is, since conjunction and negation form an expressively complete set of operators, EG is expressively complete as well (and EG does not need parentheses!)

# *Symbolization*

## **From PL to EG**

Expression in PL

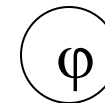
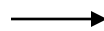
$P$



Symbolization in EG

$P$

$\sim\varphi$

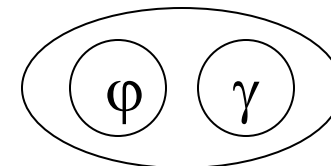
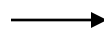


$\varphi \wedge \gamma$

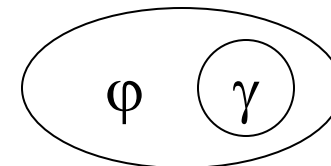


$\varphi \quad \gamma$

$\varphi \vee \gamma$



$\varphi \rightarrow \gamma$



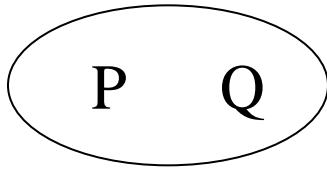
## *Symbolization*

# From EG to PL

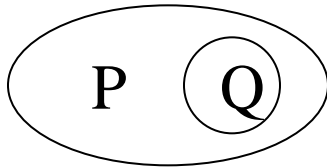
### Possible Readings

P Q

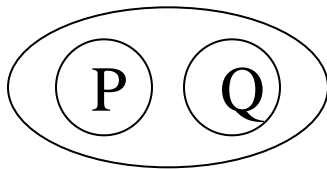
$P \wedge Q$  or  $Q \wedge P$  or P and Q



$\sim(P \wedge Q)$  or  $\sim(Q \wedge P)$  or  
 $\sim P \vee \sim Q$  or  $\sim Q \vee \sim P$



$\sim(P \wedge \sim Q)$  or  $\sim(\sim Q \wedge P)$  or  
 $\sim P \vee Q$  or  $Q \vee \sim P$  or  $P \rightarrow Q$



$\sim(\sim P \wedge \sim Q)$  or  $\sim P \rightarrow Q$  or  $P \vee Q$  or  
 $\sim(\sim Q \wedge \sim P)$  or  $\sim Q \rightarrow P$  or  $Q \vee P$

## *Inference*

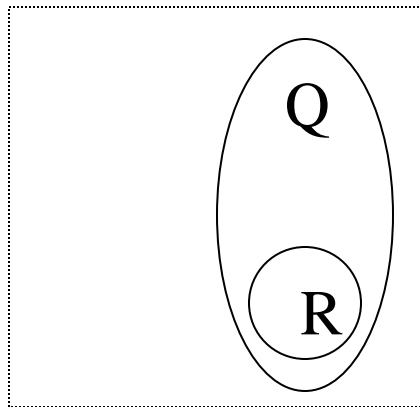
# **Inference Rules**

- Alpha has four inference rules:
  - 2 rules of inference:
    - Insertion
    - Erasure
  - 2 rules of equivalence:
    - Double Cut
    - Iteration/Deiteration
- To understand these inference rules, one first has to grasp the concepts of subgraph, double cut, level, and nested level.

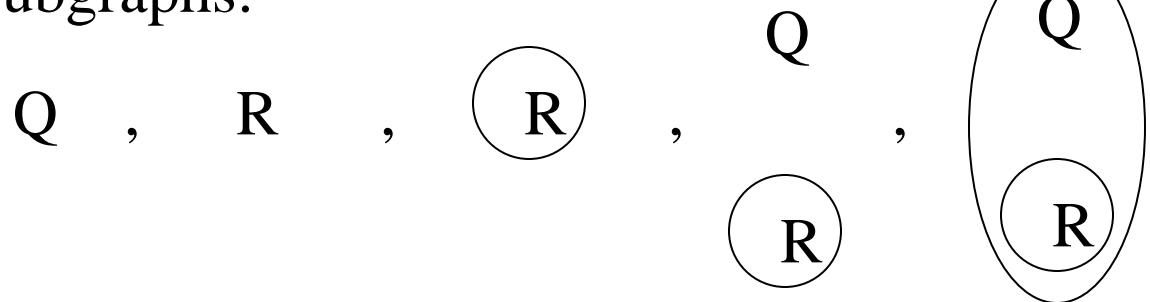
## *Inference*

# Subgraph

- The notion of subgraph is best illustrated with an example:



The graph on the left has the following subgraphs:



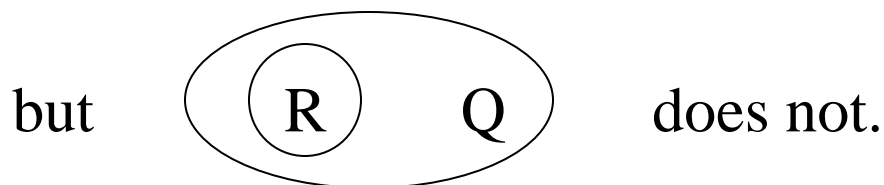
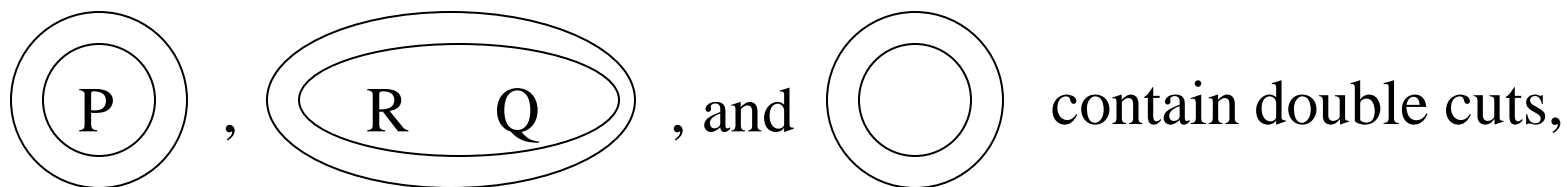
In other words, a subgraph is any part of the graph, as long as cuts keep all of their contents. Any graph is a subgraph of itself, and empty graphs can be considered subgraphs as well.



## *Inference*

# Double Cut

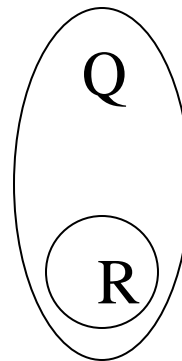
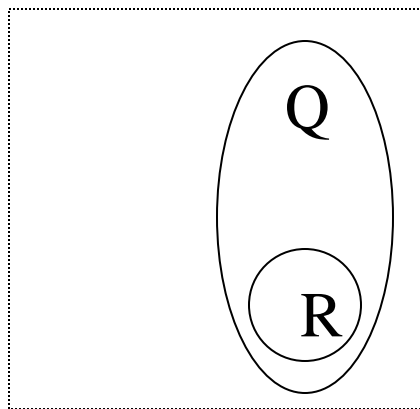
- A Double Cut is any pair of cuts where one is inside the other and where there is only the empty graph in between. Thus:






## *Inference*

# Level

- The level of any subgraph is the number of cuts around it. Thus, in the following graph:



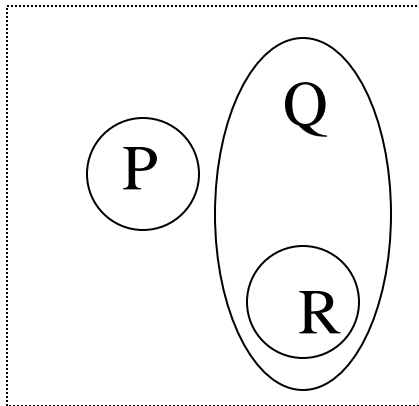
(the graph itself) is at level 0,

Q ,  , and  are at level 1, and  is at level 2


## *Inference*

# Nested Level

- A subgraph  $\varphi$  is said to exist at a *nested level* in relation to some other subgraph  $\gamma$  if and only if one can go from  $\gamma$  to  $\varphi$  by going inside zero or more cuts, and without going outside of any cuts. E.g. in the graph below:



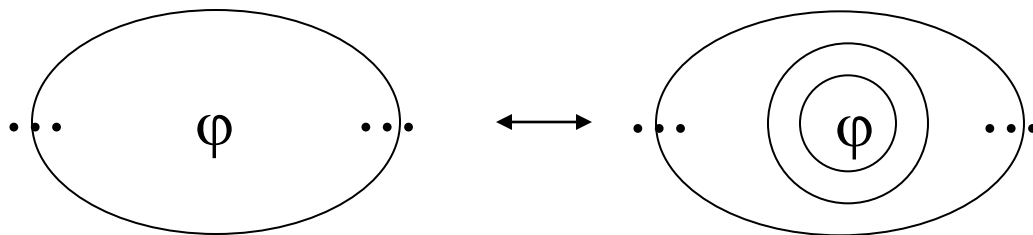
R exists at a nested level in relation to Q, but not in relation to P. Also:

Q and  exist at a nested level in relation to each other.

## *Inference*

# Double Cut

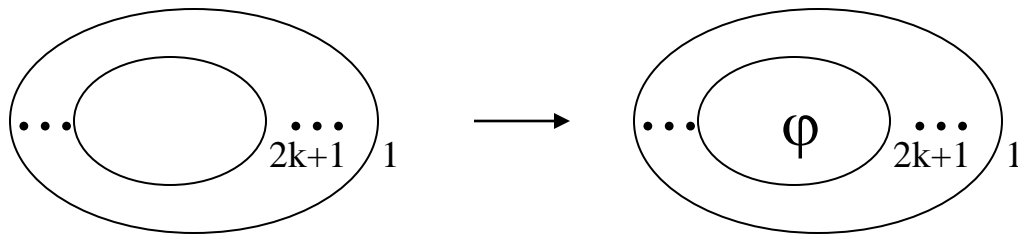
- The Double Cut rule of equivalence allows one to draw or erase a double cut around any subgraph. Obviously, this rule corresponds exactly with Double Negation from PL.



## *Inference*

# Insertion

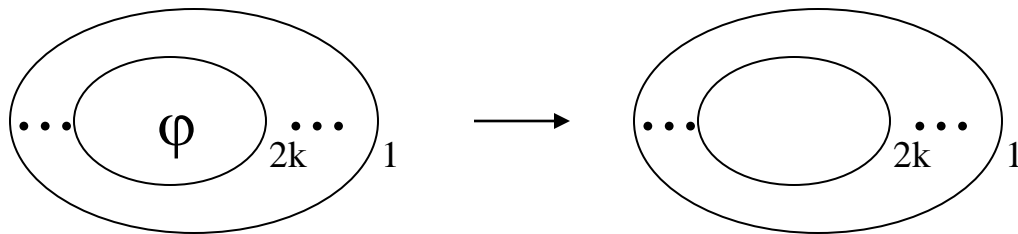
- The Insertion rule allows one to insert any graph at any odd level.



## *Inference*

# **Erasure**

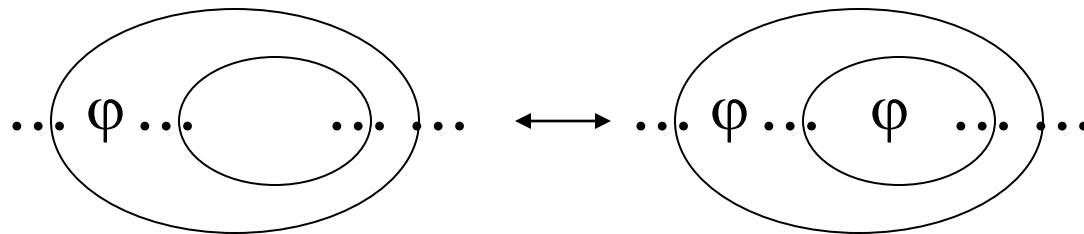
- The Erasure rule of inference allows one to erase any graph from any even level.



## *Inference*

# Iteration/Deiteration

- The Iteration/Deiteration rule of equivalence allows one to place or erase a copy of any subgraph at any nested level in relation to that subgraph.



## *Inference*

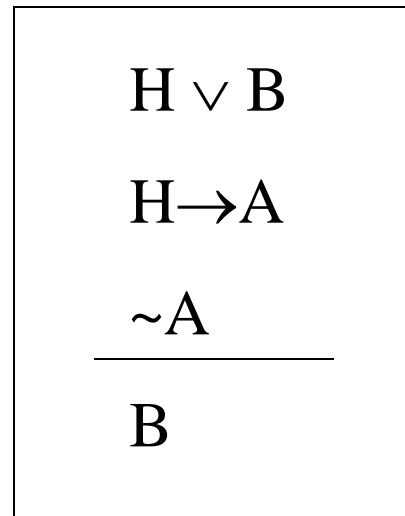
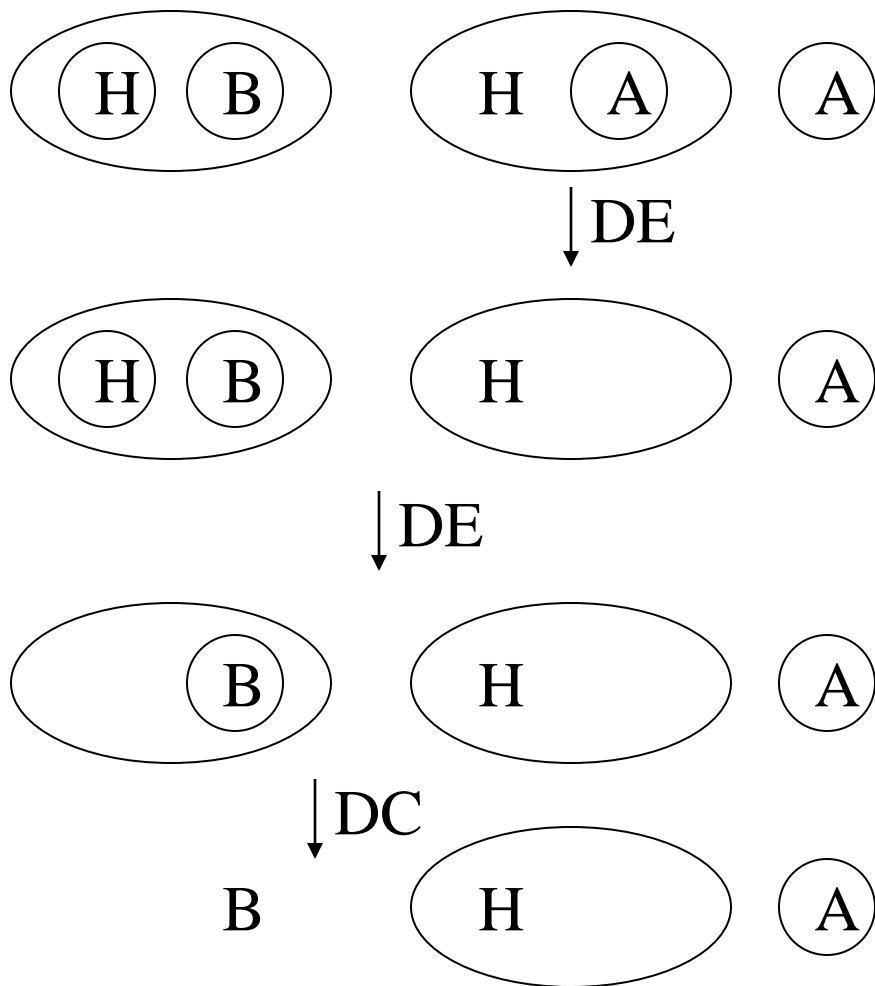
# **Formal Proofs**

- A formal proof in EG consists in the successive application of inference rules to transform one graph into another.
- Formal proofs in EG are used just as in PL:
  - To show that an argument is valid, transform the graph of the premises into the graph of the conclusion.
  - To show that a set of statements is inconsistent transform the graph of the statements into an empty cut.
  - To show that two statements are equivalent, transform the one into the other, and vice versa.
  - To show that a statement is a tautology, transform an empty graph into the graph of that statement.



*Inference*

# Sample Proof in EG



## *Inference*

# Transforming rather than Rewriting

- An interesting difference between doing formal proofs in EG and doing formal proofs in traditional systems is that in the former one *transforms* (by adding or deleting) a *single* graph, whereas in the latter one deals with *multiple* sentences, and has to do a lot of *rewriting*.
- Example: Suppose we want to infer  $Q \wedge (R \rightarrow S)$  from  $P$  and  $P \rightarrow [Q \wedge (R \rightarrow S)]$ . In PL, we would use Modus Ponens to go from two separate statements to a third, having to rewrite all of  $Q \wedge (R \rightarrow S)$  on a separate line. In EG, we have a single graph being the juxtaposition of the symbolizations of  $P$  and  $P \rightarrow [Q \wedge (R \rightarrow S)]$ , after which the second  $P$  gets deleted by deiteration and the desired result is obtained through the simple elimination of a double cut.

## *Inference*

# **Proofs as Movies**

- Because graphs are being transformed rather than rewritten, proofs in EG are going to look quite different from proofs in PL.
- Proofs become like videos that one can play, rewind, fast-forward, etc.
- It would be interesting to see if this dynamic character of proofs has any further conceptual consequences as far as people are able to do proofs and think about proofs.

## *Inference*

# **Subproofs**

- Another interesting difference between doing formal proofs in EG and PL is that in EG there is no need for doing subproofs.
- Of course, one could define subproofs in EG, but one should notice that at that point one is no longer dealing with a single graph that is being transformed: extra formal machinery is needed to deal with subproofs, just as in PL.
- The interesting fact is that the 4 inference rules of EG are both sound and complete, even though they don't use subproofs (see “Alpha: Soundness and Completeness”).

## *Inference*

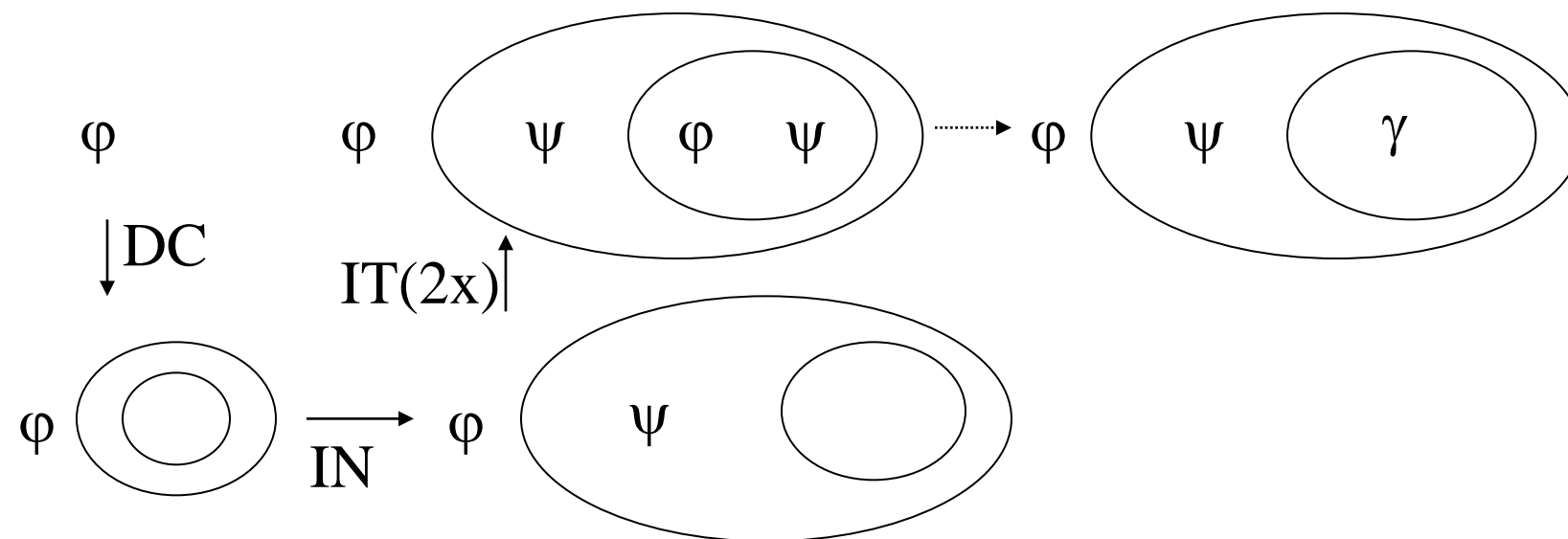
# **Simulating Subproofs**

- EG does not have subproofs. However, subproofs can be simulated using the rules of EG in the following manner:
  - 1. Draw an empty double cut on level 0.
  - 2. Insert the assumption of the subproof within the outer cut (i.e. on level 1).
  - 3. Iterate the original graph within the inner cut, as well as the extra assumption.
  - 4. Manipulate the graphs on level 2 as usual.
  - 5. Use obtained result appropriately (see next slides)
- Subproofs within subproofs can be done at levels 2, 4, etc.

## *Inference*

# Conditional Proofs

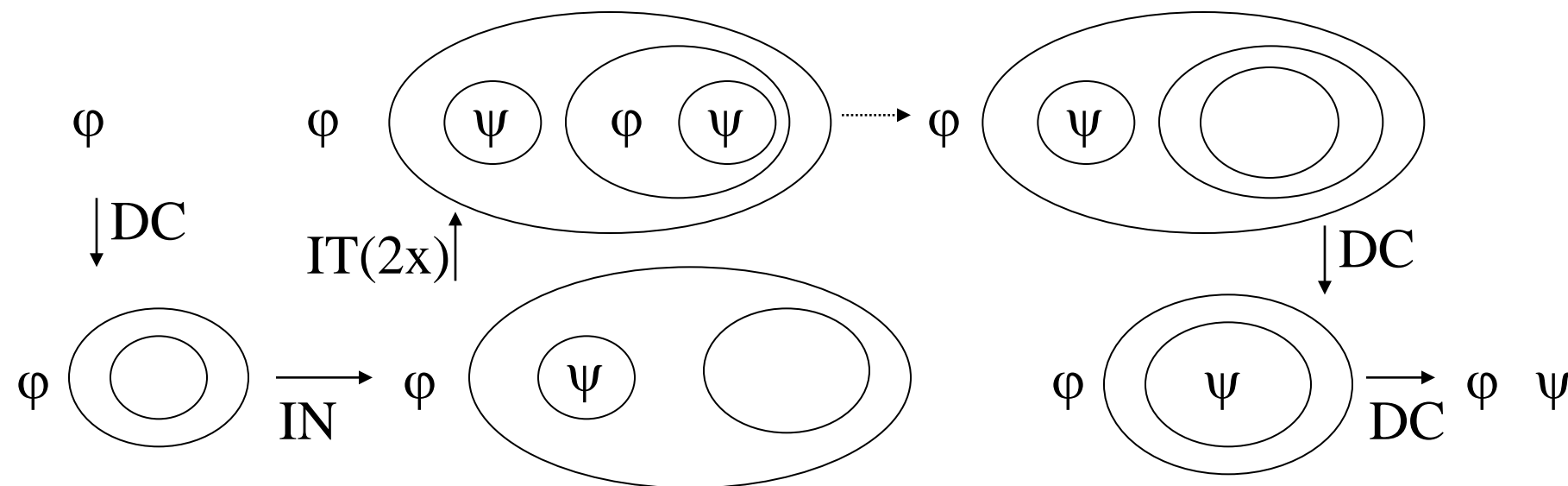
- Simulating Conditional Proof:
  - The assumption is the antecedent ( $\psi$ ) of the desired conditional
  - After iterating the original graph ( $\varphi$ ) and the assumption on the even level, one tries to derive the consequent ( $\gamma$ ).
  - The result is the desired conditional.



## Inference

# Indirect Proof

- Simulating Indirect Proof:
  - The assumption is the negation of the desired goal ( $\psi$ )
  - After iterating the original graph ( $\varphi$ ) and the assumption on the even level, one tries to derive an empty cut.
  - Once the empty cut has been obtained, the desired goal can be obtained through double cut elimination.



## *Inference*

# **Deriving Empty Cuts**

- Deriving an empty cut often merely requires the application of Erasure, Deiteration, and erasing double cuts.
- In other words, one often merely has to eliminate parts of the graph in order to derive a contradiction.



## **Efficiency of Proofs**

- In traditional PL systems, there is a trade-off between the number of inference rules and the number of steps of a formal proof: if one wants a formal proof to require fewer steps, one has to introduce more inference rules, and if one wants fewer inference rules, formal proofs will require more steps.
- While EG has fewer rules (4) than traditional PL systems (10 to 20), proofs in EG usually require fewer steps!

## *Inference*

# **Ease of Proofs**

- Although hard empirical data needs to be gathered, doing proofs in EG seems to be easier than doing proofs in PL. Possible reasons for this:
  - Graphical representation
  - Transforming rather than rewriting
  - No Subproofs
  - Fewer rules, fewer steps.
  - Ease of deriving empty cut.

# Existential Graphs Home Page

- You can read more about Existential Graphs, and play with a (somewhat) working Existential Graphs applet at:  
<http://www.cogsci.rpi.edu/~heuweb/research/EG/eg.html>
- Programmers needed!
  - Possibility of paid summer research position